

Correspondences between partitions

Roland Glantz, Christian L. Staudt, and Henning Meyerhenke

Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany

Abstract. A partition is a subdivision of a set into disjoint subsets called parts. Partitions are instrumental in tasks such as classification, pattern recognition and network analysis. In the past, a wide spectrum of similarity measures for pairs of partitions $\mathcal{P} = \{P_1, \dots, P_{|\mathcal{P}|}\}$ and $\mathcal{P}' = \{P'_1, \dots, P'_{|\mathcal{P}'|}\}$ of the same set V have been proposed. Such a measure for the overall similarity of \mathcal{P} and \mathcal{P}' , however, does not provide specifics on how \mathcal{P} and \mathcal{P}' correspond, e. g. when the union of two parts in \mathcal{P} is similar to that of three parts in \mathcal{P}' .

In this paper, we think of a good correspondence between \mathcal{P} and \mathcal{P}' as a pair $(\mathcal{S}, \mathcal{S}')$ with $\mathcal{S} \subseteq \mathcal{P}$ and $\mathcal{S}' \subseteq \mathcal{P}'$ such that the symmetric difference between the union of the parts in \mathcal{S} and the union of the parts in \mathcal{S}' has low cardinality (low total weight if the elements of V are weighted). It turns out that such an objective function is essentially one for subsets \mathcal{S} of \mathcal{P} only, and that it makes sense to call the objective function's value for \mathcal{S} the *fragmentation of \mathcal{S} w. r. t. \mathcal{P}'* or the *weight of the cut $(\mathcal{S}, \mathcal{P} \setminus \mathcal{S})$ w. r. t. \mathcal{P}'* .

Thus, we can derive a hierarchy of the $|\mathcal{P} - 1|$ best correspondences from a minimal cut basis of \mathcal{P} , i. e. $|\mathcal{P} - 1|$ minimal non-crossing P_s - P_t cuts of \mathcal{P} . Since our objective function is symmetric and submodular, we can compute a minimal P_s - P_t cut in time $\mathcal{O}(|V|) + |\mathcal{P}|^4 |\mathcal{P}'|$ for all $P_s \neq P_t \in \mathcal{P}$. To reduce the running time for typical inputs, we present a branch-and-bound algorithm for finding minimal P_s - P_t cuts, which is feasible up to $|\mathcal{P}|, |\mathcal{P}'| \approx 1000$ when the entire cut basis is to be computed. The number of elements in V is practically irrelevant.

Finally, we use correspondences to gain insight into a parallel community detection algorithm that is non-deterministic due to race conditions and that has an optional refinement phase. It turns out that the race conditions frequently entail unions or break-ups of communities, while the refinement leaves most communities intact.

Keywords: Correspondences between partitions, symmetric submodular optimization, similarity measures for partitions, minimal cuts, branch-and-bound algorithm

1 Introduction

“One of the most basic abilities of living creatures involves the grouping of similar objects ...” [4]. Objective and quantitative methods to help humans with

the task of grouping objects in a meaningful way are the subject of cluster analysis [4], e.g. when organisms are to be grouped according to their genetic similarities, when a classification is to be learned, when the pixels/voxels of an image are to be grouped such that the groups correspond to real-world objects or when communities in a social network are to be detected. Usually, the parts are non-overlapping and form a partition of data points into parts/clusters/groups/classes/cells/regions/communities.

This paper is about a new approach for comparing two partitions $\mathcal{P} = \{P_1, \dots, P_{|\mathcal{P}|}\}$ and $\mathcal{P}' = \{P'_1, \dots, P'_{|\mathcal{P}'|}\}$ of the same set. In contrast to similarity measures for partitions that yield a single number [16,27], we provide specifics on how \mathcal{P} and \mathcal{P}' correspond to each other.

More specifically, a good correspondence is a pair $(\mathcal{S}, \mathcal{S}')$ with $\mathcal{S} \subseteq \mathcal{P}$, $\mathcal{S}' \subseteq \mathcal{P}'$ and a low value of

$$\phi_{\mathcal{P}, \mathcal{P}'}(\mathcal{S}, \mathcal{S}') := |U_{\mathcal{S}} \Delta U_{\mathcal{S}'}|, \quad (1.1)$$

where $U_{\mathcal{S}}$ denotes the union of all sets in \mathcal{S} , Δ denotes the symmetric difference, and $|\cdot|$ denotes cardinality [total weight] if the elements of V are unweighted [weighted]. Thus, minimizing $\phi_{\mathcal{P}, \mathcal{P}'}(\cdot, \cdot)$ amounts to finding similarities between \mathcal{P} and \mathcal{P}' modulo unions of parts.

If \mathcal{P} and \mathcal{P}' are partitions of sets that are different but have a large intersection, W , it may be appropriate to turn \mathcal{P} and \mathcal{P}' into the two related partitions $\{P_1 \cap W, \dots, P_{|\mathcal{P}|} \cap W\}$ and $\{P'_1 \cap W, \dots, P'_{|\mathcal{P}'|} \cap W\}$. If W is large enough, the correspondences between the two new partitions will still reveal specifics on similarities between \mathcal{P} and \mathcal{P}' .

Correspondences in image segmentation. For the sake of graphic power we turn to applications in which \mathcal{P} and \mathcal{P}' are segmentations, i.e. partitions of a set of pixels/voxels into regions (actual use of correspondences in this paper will be in community detection, see Section 5.3).

We assume that \mathcal{P} and \mathcal{P}' are based on the same image or, if they are based on different images, that the latter have the same resolution and picture the same scene. Ideally, a region corresponds to a real-world object, but finding such regions is hindered by noise, under-segmentation, over-segmentation or occlusion. Scenarios in which it makes sense to compare \mathcal{P} and \mathcal{P}' using correspondences can be as follows.

1. \mathcal{P} is the result of a segmentation algorithm and \mathcal{P}' describes ground truth, e.g. if \mathcal{P} is a segmented satellite image and if \mathcal{P}' describes land use that has been determined in the field by experts. Here, the aim of a comparison might be to identify areas where \mathcal{P} suffers from over-segmentation (unions of regions of \mathcal{P} that correspond well to single regions of \mathcal{P}'), from under-segmentation (single regions of \mathcal{P} that correspond well to unions of regions of \mathcal{P}') or more intricate combinations of over-segmentation and under-segmentations, i.e. good correspondences $(\mathcal{S}, \mathcal{S}')$ with $|\mathcal{S}|, |\mathcal{S}'| > 1$. There may also be areas in which there are no good correspondences, i.e. where the segmentation algorithm fails completely.

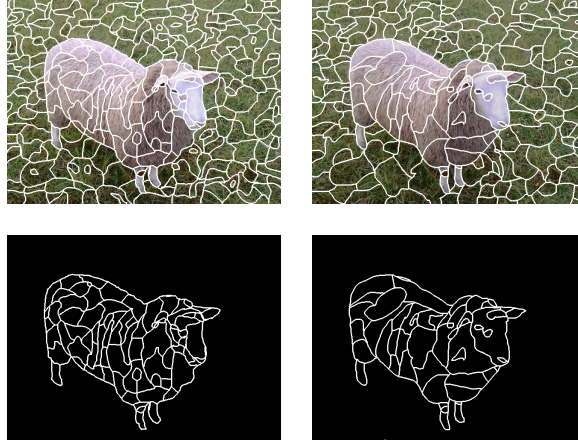


Fig. 1. Top: Two segmentations (by hand) of the same image. Poor match between individual regions left and right. Bottom: A good correspondence.

2. \mathcal{P} and \mathcal{P}' describe ground truth at different times. Sticking to land use, a good correspondence $(\mathcal{S}, \mathcal{S}')$ with $|\mathcal{S}|, |\mathcal{S}'| > 1$ may indicate crop rotation.
3. \mathcal{P} and \mathcal{P}' are results of different segmentation algorithms applied to the same image, and/or the two segmentations are based on different physical measurements, e. g. channels in Satellite Imagery or CT vs. MRI in medical imaging. Then, a good correspondence $(\mathcal{S}, \mathcal{S}')$ provides strong evidence that the feature described by \mathcal{S} is not an artifact. For an example of correspondences between different segmentations see Figure 1.

Contributions. Our first contribution is to show that minimizing $\phi_{\mathcal{P}, \mathcal{P}'}(\mathcal{S}, \mathcal{S}')$ in Equation (1.1) under the restriction $\emptyset \neq \mathcal{S} \subsetneq \mathcal{P}$ essentially amounts to minimizing a function that does not depend on \mathcal{S}' anymore. In particular, it turns out that

$$\phi_{\mathcal{P}, \mathcal{P}'}(\mathcal{S}) := \min_{\mathcal{S}' \subseteq \mathcal{P}'} \phi_{\mathcal{P}, \mathcal{P}'}(\mathcal{S}, \mathcal{S}') \quad (1.2)$$

$$= \sum_{P' \in \mathcal{P}'} |P'| \text{peak}\left(\frac{|U_{\mathcal{S}} \cap P'|}{|P'|}\right), \quad \text{where} \quad (1.3)$$

$$\text{peak}(x) = \begin{cases} x, & \text{if } x \leq 1/2, \\ 1 - x, & \text{if } x > 1/2. \end{cases} \quad (1.4)$$

The value $\phi_{\mathcal{P}, \mathcal{P}'}(\mathcal{S})$ can be seen as the *fragmentation of \mathcal{S} w. r. t. \mathcal{P}'* or as the *weight of the cut $(\mathcal{S}, \mathcal{P} \setminus \mathcal{S})$ w. r. t. \mathcal{P}'* . An optimal partner \mathcal{S}' of \mathcal{S} , see Equation (1.2), then is

$$\mathcal{S}' = \{P' \in \mathcal{P}' : |U_{\mathcal{S}} \cap P'| > \frac{|P'|}{2}\}. \quad (1.5)$$

Thus, we reduce the problem of finding good correspondences $(\mathcal{S}, \mathcal{S}')$ to the problem of finding $\emptyset \neq \mathcal{S} \subsetneq \mathcal{P}$ that give rise to small cuts $(\mathcal{S}, \mathcal{P} \setminus \mathcal{S})$. It turns out that $\phi_{\mathcal{P}, \mathcal{P}'}(\cdot)$ is a symmetric submodular function, see Section 3.1.

Using results from symmetric submodular minimization, our second contribution consists in deriving an asymptotic time and space complexity for minimizing fragmentation under the constraint that \mathcal{S} is nontrivial. Specifically, we show that $\emptyset \neq \mathcal{S} \subsetneq \mathcal{P}$ minimizing $\phi_{\mathcal{P}, \mathcal{P}'}(\cdot)$ can be found in time $\mathcal{O}(|V| + |\mathcal{P}|^4 |\mathcal{P}'|)$, where V is the ground set of \mathcal{P} and \mathcal{P}' . This is done in Sections 3.2 and 3.3.

To make the detection of good correspondences practical, our third contribution is an efficient method to compute a basis of minimal P_s - P_t cuts of \mathcal{P} w. r. t. \mathcal{P}' . This cut basis gives rise to the $|\mathcal{P}| - 1$ best correspondences via Equations (1.2) and (1.5). We first show that finding such a basis requires $|\mathcal{P}| - 1$ computations of certain minimal P_s - P_t cuts. We then present a branch-and-bound algorithm to compute a minimal P_s - P_t cut for any pair $P_s \neq P_t$, see Section 4.2. We test the performance of our algorithm in Section 5.2. As an example, computing the $|\mathcal{P}| - 1$ best correspondences between two partitions with 100 parts each takes only 11 seconds, even if the best correspondence is well hidden and involves large $\mathcal{S}, \mathcal{S}'$.

Our fourth contribution is to the field of community detection. Typically, a community is a densely connected set of nodes with sparse connections to the rest of the network (for surveys on methods to find communities see [19, 5]). In Section 5.3 we use correspondences to gain insight into the behavior of a community detection algorithm [3] and an extension thereof [22]. Specifically, we compare two effects on the resulting partitions: (i) the effect of race conditions that occur during the parallel execution of the algorithm and (ii) the effect of an optional refinement phase. It turns out that the overall quality of the correspondences due to these effects is on the same order, that the race conditions lead to many unions and break-ups of communities, but that the refinement phase leaves most communities intact.

2 Examples of correspondences and cuts

2.1 Correspondences

Figure 2 depicts two partitions of a set V with 31 elements. Specifically, we have a partition $\mathcal{P} = \{P_1, \dots, P_6\}$ on top and a partition $\mathcal{P}' = \{P'_1, \dots, P'_5\}$ at the bottom. The elements of V are represented by symbols indicating membership to the parts of \mathcal{P} . The four nontrivial subsets of \mathcal{P} with minimum fragmentation (value is 2) are the subsets $\{P_1, P_2\}$ and $\{P_5, P_6\}$, as well as the complements of these subsets. The best partners (see Equation (1.5)) of the former two subsets are the subsets $\{P'_1\}$ and $\{P'_4, P'_5\}$, giving rise to the correspondences $(\{P_1, P_2\}, \{P'_1\})$ and $(\{P_5, P_6\}, \{P'_4, P'_5\})$, respectively (dissimilarity is 2 in both cases). The best partner of $\{P_4\}$ is $\{P'_3\}$.

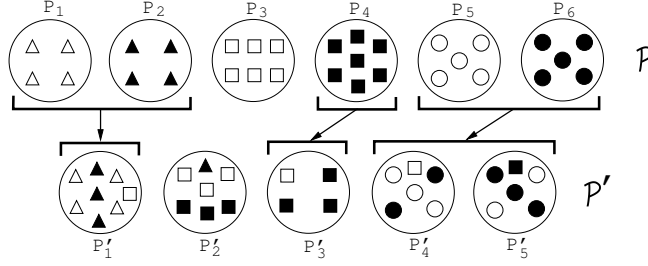


Fig. 2. Upper [lower] row of six [five] disks depicts partition \mathcal{P} [\mathcal{P}']. Upper [lower] brackets indicate subsets of \mathcal{P} [\mathcal{P}'], and arrows indicate some good correspondences between subsets of \mathcal{P} and those of \mathcal{P}' (see the text).

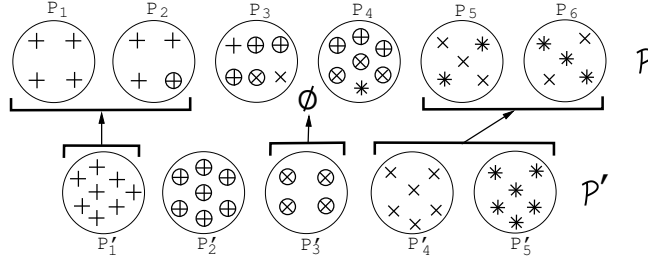


Fig. 3. Same scenario as in Figure 2 with the roles of \mathcal{P} and \mathcal{P}' exchanged.

If we reverse the roles of \mathcal{P} and \mathcal{P}' (see Figure 3), the counterparts of the best nontrivial correspondences from before are the new best nontrivial correspondences, e. g. $(\{P'_1\}, \{P_1, P_2\})$, and $(\{P'_4, P'_5\}, \{P_5, P_6\})$. The counterpart of the correspondence $(\{P_4\}, \{P'_3\})$, however, is gone. Indeed, the best partner of $\mathcal{S}' = \{P'_3\}$ is not $\{P_4\}$ but \emptyset . In particular, $|P'_3 \triangle \emptyset| = 4 < 5 = |P'_3 \triangle P_4|$. Hence, we should be aware that reversing the roles of \mathcal{P} and \mathcal{P}' cannot always be compensated by swapping \mathcal{S} and \mathcal{S}' in a correspondence.

2.2 Basis of minimal P_s - P_t cuts

The $|\mathcal{P}| - 1$ best correspondences that we calculate in this paper can be derived from a basis of minimal P_s - P_t cuts. Specifically, a minimal P_s - P_t cut $(\mathcal{S}, \mathcal{P} \setminus \mathcal{S})$ from the basis gives rise to the correspondence $(\mathcal{S}, \mathcal{S}')$, where \mathcal{S}' is determined by Equation (1.5).

Such a basis of minimal P_s - P_t cuts, in turn, can be represented by an edge-weighted tree called Gomory-Hu tree [10]. For an exact definition of Gomory-Hu trees and proof of the statements below see Definition 5 and Proposition 5. For examples of Gomory-Hu trees see Figures 4a,c. A Gomory-Hu tree is a very concise representation of *all* minimal P_s - P_t cuts and, in conjunction with Equation (1.5), gives rise to the entire hierarchy of the $|\mathcal{P}| - 1$ best correspondences.

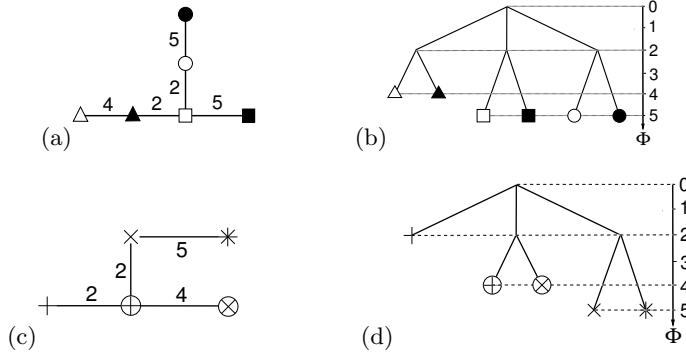


Fig. 4. \mathcal{P} and \mathcal{P}' are as in Figures 2 and 3. (a) Gomory-Hu tree of best cuts of \mathcal{P} w.r.t. \mathcal{P}' and (b) hierarchical decomposition of \mathcal{P} . (c) Gomory-Hu tree of best cuts of \mathcal{P}' w.r.t. \mathcal{P} and (d) hierarchical decomposition of \mathcal{P}' .

The nodes of a Gomory-Hu tree $T_{\mathcal{P}|\mathcal{P}'}$ represent the parts of \mathcal{P} (see Figures 4a,c.), and the edges of $T_{\mathcal{P}|\mathcal{P}'}$ represent minimal P_s - P_t cuts in the following way. Removal of an edge $\{P_s, P_t\}$ from $T_{\mathcal{P}|\mathcal{P}'}$ results in two subtrees, one spanning a subset \mathcal{S} of \mathcal{P} , and the other spanning $\mathcal{P} \setminus \mathcal{S}$. This defines the P_s - P_t cut $(\mathcal{S}, \mathcal{P} \setminus \mathcal{S})$ that is always a *minimal* P_s - P_t cut. As an example, removal of the edge between \blacktriangle and \square in Figure 4a gives rise to the cut that separates the triangles from the other shapes. This cut is a minimal \blacktriangle - \square cut.

The edge weights of $T_{\mathcal{P}|\mathcal{P}'}$ are such that for any $P_s \neq P_t \in \mathcal{P}$, not necessarily connected by an edge of $T_{\mathcal{P}|\mathcal{P}'}$, we have that (i) an edge on the unique path from P_s to P_t with minimal weight w gives rise to a minimal P_s - P_t cut, and (ii) the weight of this cut is w . As an example, a minimal P_1 - P_4 cut of \mathcal{P} , as shown in Figure 2, is given by the edge between \blacktriangle and \square in Figure 4a, and the weight of this cut is 2.

The Gomory-Hu tree of the $|\mathcal{P}| - 1 = 5$ best cuts of \mathcal{P} is shown in Figure 4a. A dendrogram of the hierarchical decomposition of \mathcal{P} , as determined by the five best cuts, is shown in Figure 4b. Figures 4c,d are the corresponding figures for \mathcal{P}' instead of \mathcal{P} .

3 Finding an optimal correspondence

3.1 Correspondences from fragmentation minimization

Recall that the union of the parts in some $\mathcal{S} \subseteq \mathcal{P}$ is denoted by $U_{\mathcal{S}}$. We want to minimize $\phi_{\mathcal{P}, \mathcal{P}'}(\mathcal{S}, \mathcal{S}') = |U_{\mathcal{S}} \Delta U_{\mathcal{S}'}|$, where $\mathcal{S} \subseteq \mathcal{P}$ and $\mathcal{S}' \subseteq \mathcal{P}'$.

The solutions $(\mathcal{S}, \mathcal{S}')$ with $\mathcal{S} \in \{\emptyset, \mathcal{P}\} \wedge \mathcal{S}' \in \{\emptyset, \mathcal{P}'\}$, however, are not useful and should be excluded. We first exclude even more solutions, but we will make up for this below when we exchange the roles of \mathcal{P} and \mathcal{P}' .

Definition 1 (Eligible pair $(\mathcal{S}, \mathcal{S}')$, optimal pair $(\mathcal{S}, \mathcal{S}')$). A pair $(\mathcal{S}, \mathcal{S}')$ with $\mathcal{S} \notin \{\emptyset, \mathcal{P}\}$ is called eligible for $\phi_{\mathcal{P}, \mathcal{P}'}(\cdot, \cdot)$. An eligible pair $(\mathcal{S}, \mathcal{S}')$ is called optimal for $\phi_{\mathcal{P}, \mathcal{P}'}(\cdot, \cdot)$ if $\phi_{\mathcal{P}, \mathcal{P}'}(\mathcal{S}, \mathcal{S}')$ is minimal w. r. t. all eligible pairs.

Exchanging the roles of \mathcal{P} and \mathcal{P}' (compare Figures 2 and 3) yields a different optimization problem since eligibility is not symmetric. Now, among all useful pairs, i. e. pairs $(\mathcal{S}, \mathcal{S}')$ with $\mathcal{S} \notin \{\emptyset, \mathcal{P}\} \vee \mathcal{S}' \notin \{\emptyset, \mathcal{P}'\}$, a pair $(\mathcal{S}, \mathcal{S}')$ minimizes $|U_{\mathcal{S}} \Delta U_{\mathcal{S}'}|$ if and only if $(\mathcal{S}, \mathcal{S}')$ is an optimal pair for $\phi_{\mathcal{P}, \mathcal{P}'}(\cdot, \cdot)$ or $(\mathcal{S}', \mathcal{S})$ is an optimal pair for $\phi_{\mathcal{P}', \mathcal{P}}(\cdot, \cdot)$. The following results on $\phi_{\mathcal{P}, \mathcal{P}'}(\mathcal{S}, \mathcal{S}')$ yield corresponding results on $\phi_{\mathcal{P}', \mathcal{P}}(\mathcal{S}', \mathcal{S})$.

Proposition 1 (From $\phi_{\mathcal{P}, \mathcal{P}'}(\cdot, \cdot)$ to $\phi_{\mathcal{P}, \mathcal{P}'}(\cdot, \cdot)$). An optimal pair $(\mathcal{S}, \mathcal{S}')$ of $\phi_{\mathcal{P}, \mathcal{P}'}(\cdot, \cdot)$ can be found by first finding $\emptyset \neq \mathcal{S} \subsetneq \mathcal{P}$ that minimizes

$$\phi_{\mathcal{P}, \mathcal{P}'}(\mathcal{S}) := \sum_{P' \in \mathcal{P}'} |P'| \text{peak}\left(\frac{|U_{\mathcal{S}} \cap P'|}{|P'|}\right), \quad \text{where} \quad (3.1)$$

$$\text{peak}(x) := \begin{cases} x, & \text{if } x \leq 1/2 \\ 1 - x, & \text{if } x > 1/2 \end{cases}. \quad (3.2)$$

and then setting

$$\mathcal{S}' := \{P' \in \mathcal{P}' : |U_{\mathcal{S}} \cap P'| > \frac{|P'|}{2}\}. \quad (3.3)$$

Proof: Starting with Equation (1.1), we get

$$\phi_{\mathcal{P}, \mathcal{P}'}(\mathcal{S}, \mathcal{S}') = |U_{\mathcal{S}} \setminus U_{\mathcal{S}'}| + |U_{\mathcal{S}'} \setminus U_{\mathcal{S}}| \quad (3.4)$$

$$= |U_{\mathcal{S}} \cap (V \setminus U_{\mathcal{S}'})| + \sum_{P' \in \mathcal{S}'} |P' \setminus U_{\mathcal{S}}| \quad (3.5)$$

$$= \sum_{P' \notin \mathcal{S}'} |U_{\mathcal{S}} \cap P'| + \sum_{P' \in \mathcal{S}'} (|P'| - |U_{\mathcal{S}} \cap P'|). \quad (3.6)$$

By choosing \mathcal{S}' as in Equation (3.3), we minimize the contribution (damage) of each $P' \in \mathcal{P}'$ to the right hand side of Equation (3.6), and thus minimize $\phi_{\mathcal{P}, \mathcal{P}'}(\mathcal{S}, \cdot)$. Insertion of \mathcal{S}' from Equation (3.3) then yields

$$\min_{\mathcal{S}' \subseteq \mathcal{P}'} \phi_{\mathcal{P}, \mathcal{P}'}(\mathcal{S}, \mathcal{S}') = \sum_{P' \in \mathcal{P}'} \min\{|U_{\mathcal{S}} \cap P'|, |P'| - |U_{\mathcal{S}} \cap P'|\} \quad (3.7)$$

$$= \sum_{P' \in \mathcal{P}'} |P'| \min\left\{\frac{|U_{\mathcal{S}} \cap P'|}{|P'|}, 1 - \frac{|U_{\mathcal{S}} \cap P'|}{|P'|}\right\} \quad (3.8)$$

$$= \sum_{P' \in \mathcal{P}'} |P'| \text{peak}\left(\frac{|U_{\mathcal{S}} \cap P'|}{|P'|}\right). \quad (3.9)$$

□

Recall that we refer to $\phi_{\mathcal{P},\mathcal{P}'}(\mathcal{S})$ as the *fragmentation of \mathcal{S} w. r. t. \mathcal{P}'* or the *weight of the cut $(\mathcal{S}, \mathcal{P} \setminus \mathcal{S})$ w. r. t. \mathcal{P}'* .

The function $\text{peak}(\cdot)$ in Equation (3.2), called (classification) error in [26], is an example of a *generator* as defined in [21]: a function $f : [0, 1] \mapsto \mathbb{R}$ is a generator if it is concave and $f(0) = f(1) = 0$ (hence $f(\cdot)$ is also subadditive). In addition, $\text{peak}(\cdot)$ is symmetric, i. e. $\text{peak}(p) = \text{peak}(1-p)$ for all $p \in [0, 1]$. Other examples of symmetric generators are the binary entropy function $H(\cdot)$ [15,26,21] and the Gini impurity measure $G(\cdot)$ [26,21]. We could have chosen $H(\cdot)$, $G(\cdot)$ or any other nontrivial symmetric generator (computable in constant time) instead of $\text{peak}(\cdot)$. The minimization of Equation (3.1) would then have the same asymptotic time complexity (see Section 3.3). We choose $\text{peak}(\cdot)$ because it is in line with minimizing $|U_{\mathcal{S}} \Delta U_{\mathcal{S}'}|$.

3.2 Fragmentation minimization and submodularity

Definition 2 (Symmetric, submodular, modular). *Let \mathcal{P} be a set. A function $\phi : 2^{\mathcal{P}} \mapsto \mathbb{R}$ is called symmetric if $\phi(\mathcal{S}) = \phi(\mathcal{P} \setminus \mathcal{S})$ for all $\mathcal{S} \subseteq \mathcal{P}$. Furthermore, $\phi(\cdot)$ is called submodular if*

$$\phi_{\mathcal{P},\mathcal{P}'}(\mathcal{S}_1 \cup \mathcal{S}_2) \leq \phi_{\mathcal{P},\mathcal{P}'}(\mathcal{S}_1) + \phi_{\mathcal{P},\mathcal{P}'}(\mathcal{S}_2) - \phi_{\mathcal{P},\mathcal{P}'}(\mathcal{S}_1 \cap \mathcal{S}_2) \quad \forall \mathcal{S}_1, \mathcal{S}_2 \subseteq \mathcal{P}.$$

If $\phi(\cdot)$ fulfills the above with “=” instead of “ \leq ”, then $\phi(\cdot)$ is called modular.

Proposition 2. *$\phi_{\mathcal{P},\mathcal{P}'}(\cdot)$ in Equation (3.1) is symmetric and submodular.*

Proof: The symmetry of $\phi_{\mathcal{P},\mathcal{P}'}(\cdot)$ follows from that of $\text{peak}(\cdot)$. Sums and multiples of submodular functions are submodular [20]. Thus, to show that $\phi_{\mathcal{P},\mathcal{P}'}(\cdot)$ is submodular, it suffices to show that $\phi_i(\mathcal{S}) := \text{peak}(\frac{|U_{\mathcal{S}} \cap P'_i|}{|P'_i|})$ in Equation (3.1) is submodular for all i .

Indeed, the $\phi_i(\cdot)$ are of the form $c(m(\cdot))$, where $c(\cdot)$ is concave and $m(\cdot)$ is non-negative modular. Any function of this form is submodular [24,2]. \square

3.3 Asymptotic running time for finding an optimal pair

In order to efficiently compute the terms $|U_{\mathcal{S}} \cap P'|$ in Equations (3.1) and (3.3), we first compute the columns of the contingency table of \mathcal{P} and \mathcal{P}' , i. e. the table (matrix) whose entry at (i, j) equals $|P_i \cap P'_j|$. We refer to these columns as distributions over \mathcal{P} .

Definition 3 (Distributions $d_{P'}[\cdot]$). *Let $P' \in \mathcal{P}'$. The distribution of P' w. r. t. \mathcal{P} is the vector $d_{P'}[\cdot]$ of length $|\mathcal{P}|$ defined by*

$$d_{P'}[i] := |P_i \cap P'| \text{ for } 1 \leq i \leq |\mathcal{P}|. \quad (3.10)$$

Proposition 3. *Calculation of all distributions can be done in time $\mathcal{O}(|\mathcal{P}||\mathcal{P}'| + |V|)$ and with space $\mathcal{O}(|\mathcal{P}||\mathcal{P}'|)$. Furthermore, given all distributions and $\mathcal{S} \subseteq \mathcal{P}$, the determination of \mathcal{S}' , as defined in Equation (3.3), and the evaluation of $\phi_{\mathcal{P},\mathcal{P}'}(\mathcal{S})$, as defined in Equation (3.1), can both be done in time $\mathcal{O}(|\mathcal{P}||\mathcal{P}'|)$ and with space $\mathcal{O}(|\mathcal{P}'|)$.*

Proof: We first show that all distributions $d_{P'}[\cdot]$ can be computed in time $\mathcal{O}(|\mathcal{P}||\mathcal{P}'| + |V|)$. Indeed, the first summand is due to initializing all $d_{P'}[\cdot]$ to zero vectors, and the second summand is due to updating the distributions, which can be done in one traversal of V (deciding on membership of any $v \in V$ to a part in \mathcal{P} and \mathcal{P}' takes constant time). The asymptotic space requirement for calculating all distributions equals that for storing the distributions, i. e. $\mathcal{O}(|\mathcal{P}||\mathcal{P}'|)$.

Using the distributions we can compute, for any given $\mathcal{S} \subseteq \mathcal{P}$, the values of all $|U_{\mathcal{S}} \cap P'|$, $P' \in \mathcal{P}'$, in time $\mathcal{O}(|\mathcal{P}||\mathcal{P}'|)$ (due to $|U_{\mathcal{S}} \cap P'| = \sum_{i: P_i \in \mathcal{S}} d_{P'}[i]$). The space requirement for this step is $\mathcal{O}(|\mathcal{P}'|)$. Based on the values of $|U_{\mathcal{S}} \cap P'|$, the determination of \mathcal{S}' via Equation 3.3, as well as the evaluation of $\phi_{\mathcal{P}, \mathcal{P}'}(\mathcal{S}, \mathcal{S}')$ via Equation (3.6), takes time and space $\mathcal{O}(|\mathcal{P}'|)$.

Recall that $\phi_{\mathcal{P}, \mathcal{P}'}(\mathcal{S}, \mathcal{S}')$ in Equation (3.6) equals $\phi_{\mathcal{P}, \mathcal{P}'}(\mathcal{S})$ in Equation (3.1). Thus, given the distributions and some $\mathcal{S} \subseteq \mathcal{P}$, the total time and space requirements for evaluating $\phi_{\mathcal{P}, \mathcal{P}'}(\mathcal{S})$ are $\mathcal{O}(|\mathcal{P}||\mathcal{P}'|)$ and $\mathcal{O}(|\mathcal{P}'|)$, respectively. \square

In the proofs of Propositions 4 and 5 we refer to Algorithm OPTIMAL-SET from [17], where a symmetric submodular function $f(\cdot)$ is minimized by building the set minimizing $f(\cdot)$ from scratch. OPTIMAL-SET consists of $\mathcal{O}(|\mathcal{P}|^3)$ evaluations of $\phi_{\mathcal{P}, \mathcal{P}'}(\cdot)$, see Theorem 3 in [17].

Proposition 4. *Finding an optimal pair $(\mathcal{S}, \mathcal{S}')$ takes time $\mathcal{O}(|V|) + |\mathcal{P}|^4|\mathcal{P}'|$ and space $\mathcal{O}(|\mathcal{P}||\mathcal{P}'|) + \mathcal{O}(|\mathcal{P}|^2)$.*

Proof: Due to Proposition 2 of this paper, Theorem 3 in [17] (which uses OPTIMAL-SET) and Proposition 3 of this paper, the asymptotic running time for minimizing Equation (3.1) with the constraint $\mathcal{S} \notin \{\emptyset, \mathcal{P}\}$ amounts to $\mathcal{O}(|V| + |\mathcal{P}|^3|\mathcal{P}||\mathcal{P}'|) = \mathcal{O}(|V| + |\mathcal{P}|^4|\mathcal{P}'|)$. The space requirement in [17] is $\mathcal{O}(|\mathcal{P}|^2)$ (needed to store the candidate subsets in Algorithm OPTIMAL-SET). \square

4 Minimal P_s - P_t cuts

We are not only interested in the best correspondence between \mathcal{P} and \mathcal{P}' , but in a larger set of *good* correspondences. Proposition 1 says that finding good correspondences basically amounts to finding small cuts of \mathcal{P} w. r. t. \mathcal{P}' . A natural basis of small cuts is formed by the minimal P_s - P_t cuts defined in the following subsection.

4.1 Basis of minimal P_s - P_t cuts

Definition 4 ((Minimal) P_s - P_t cut $(\mathcal{S}_s, \mathcal{S}_t)$, weight $\lambda_{s,t}$). *Let $P_s \neq P_t \in \mathcal{P}$. Any pair $(\mathcal{S}_s, \mathcal{S}_t)$ with $P_s \in \mathcal{S}_s$, $P_t \in \mathcal{S}_t$ and $\mathcal{S}_t = \mathcal{P} \setminus \mathcal{S}_s$ is called a P_s - P_t cut of \mathcal{P} . A P_s - P_t cut $(\mathcal{S}_s, \mathcal{S}_t)$ is minimal if $\phi_{\mathcal{P}, \mathcal{P}'}(\mathcal{S}_s)$, and thus $\phi_{\mathcal{P}, \mathcal{P}'}(\mathcal{S}_t)$, is minimal w. r. t. all P_s - P_t cuts. Finally, for any $s \neq t \in \{1, \dots, |\mathcal{P}|\}$ we set*

$$\lambda_{s,t} := \min\{\phi_{\mathcal{P}, \mathcal{P}'}(\mathcal{S}_s) : (\mathcal{S}_s, \mathcal{P} \setminus \mathcal{S}_s) \text{ is a } P_s\text{-}P_t \text{ cut}\}. \quad (4.1)$$

We are interested in minimal P_s - P_t cuts for all $s \neq t \in \{1, \dots, |\mathcal{P}|\}$. Analogous to graphs, such a set of minimal cuts can be represented by a Gomory-Hu tree [10].

Definition 5 (Gomory-Hu tree $T_{\mathcal{P}|\mathcal{P}'}$). $T_{\mathcal{P}|\mathcal{P}'} = (V_T, E_T, \omega_\phi)$.

$T_{\mathcal{P}|\mathcal{P}'} = (V_T, E_T, \omega_\phi)$ is an edge weighted tree with vertex set $V_T = \{1, \dots, |\mathcal{P}|\}$, edge set E_T and a function $\omega_\phi : E_T \mapsto \mathbb{R}_{\geq 0}$ such that for all $s \neq t \in V_T$

$$\lambda_{s,t} = \min\{\omega_\phi(e) : e \in \Pi_{s,t}\}, \text{ where} \quad (4.2)$$

$\Pi_{s,t}$ is the set of edges forming the unique path from s to t on $T_{\mathcal{P}|\mathcal{P}'}$.

Proposition 5. *There exists at least one Gomory-Hu tree $T_{\mathcal{P}|\mathcal{P}'}$, and such a Gomory-Hu tree can be computed in $\mathcal{O}(|V| + |\mathcal{P}|^5|\mathcal{P}'|)$.*

Proof: As shown in [8], the existence of $T_{\mathcal{P}|\mathcal{P}'}$ follows from $\phi_{\mathcal{P},\mathcal{P}'}(\cdot)$ being symmetric and submodular. A Gomory-Hu tree $T_{\mathcal{P}|\mathcal{P}'}$ can be built by computing $|\mathcal{P}| - 1$ minimal P_s - P_t cuts, and this is also what determines the asymptotic running time for building a Gomory-Hu tree [9]. To compute a minimal P_s - P_t cut we may again use Algorithm OPTIMAL-SET [17]. The only difference is that we require s in the set and t to be in the complement of the set. This does not change the asymptotic running time of OPTIMAL-SET.

Recall that the asymptotic running times from Propositions 3 and 4 are based on the use of OPTIMAL-SET. Thus, considering that we need to compute the distributions $d_{\mathcal{P}'}[i]$, $1 \leq i \leq |\mathcal{P}|$, only once, it follows that we can compute $T_{\mathcal{P}|\mathcal{P}'}$ in $\mathcal{O}(|V| + (|\mathcal{P}| - 1)(|\mathcal{P}|^4|\mathcal{P}'|)) = \mathcal{O}(|V| + |\mathcal{P}|^5|\mathcal{P}'|)$. \square

A Gomory-Hu tree $T_{\mathcal{P}|\mathcal{P}'}$ gives rise to a hierarchical partition of \mathcal{P} if edges from $T_{\mathcal{P}|\mathcal{P}'}$ are removed in ascending order of $\omega_\phi(\cdot)$. For examples of dendrograms depicting such hierarchical partitions see Figure 4b,d.

4.2 Finding minimal P_s - P_t cuts by branch-and-bound

Using OPTIMAL-SET, the computation of one minimal P_s - P_t cut takes time $\mathcal{O}(|V| + |\mathcal{P}|^4|\mathcal{P}'|)$, see Proposition 4. This is problematic for high $|\mathcal{P}|$. The following branch-and-bound algorithm for finding a minimal P_s - P_t cut turns out to be practical, although we cannot give guarantees on its running time.

Let $P_s \neq P_t \in \mathcal{P}$. The idea behind our algorithm to find a minimal P_s - P_t cut $(\mathcal{S}_s, \mathcal{S}_t)$ is to first set $\mathcal{S}_s := \{P_s\}$, $\mathcal{S}_t := \{P_t\}$ and then let \mathcal{S}_s and \mathcal{S}_t compete for the remaining parts in \mathcal{P} until $\mathcal{S}_s \dot{\cup} \mathcal{S}_t = \mathcal{P}$. We assure that $\mathcal{S}_s \cap \mathcal{S}_t = \emptyset$ at all times. To curtail the exponentially growing number of possibilities that arise when assigning new parts, i.e. parts in $\mathcal{P} \setminus (\mathcal{S}_s \dot{\cup} \mathcal{S}_t)$, to either \mathcal{S}_s or \mathcal{S}_t , we need a bound $b(\mathcal{S}_s \dot{\cup} \mathcal{S}_t)$ on how low $\phi_{\mathcal{P},\mathcal{P}'}(\mathcal{S})$ can possibly get for \mathcal{S} with $\mathcal{S}_s \subseteq \mathcal{S}$ and $\mathcal{S} \cap \mathcal{S}_t = \emptyset$. Proposition 6 below guarantees that the bound defined next is admissible.

Definition 6. *Let $\mathcal{S}_s, \mathcal{S}_t \subseteq \mathcal{P}$ with $P_s \in \mathcal{S}_s$, $P_t \in \mathcal{S}_t$ and $\mathcal{S}_s \cap \mathcal{S}_t = \emptyset$. We set*

$$b(\mathcal{S}_s, \mathcal{S}_t) := \sum_{P' \in \mathcal{P}'} \min\{|U_{\mathcal{S}_s} \cap P'|, |U_{\mathcal{S}_t} \cap P'|\}. \quad (4.3)$$

Proposition 6. *Let $\mathcal{S}_s, \mathcal{S}_t \subseteq \mathcal{P}$ with $P_s \in \mathcal{S}_s$, $P_t \in \mathcal{S}_t$ and $\mathcal{S}_s \cap \mathcal{S}_t = \emptyset$. Furthermore, let $\mathcal{S} \supseteq \mathcal{S}_s$, $\mathcal{S} \cap \mathcal{S}_t = \emptyset$. Then, $b(\mathcal{S}_s, \mathcal{S}_t) \leq \phi_{\mathcal{P}, \mathcal{P}'}(\mathcal{S})$.*

Proof:

$$\begin{aligned}
\phi_{\mathcal{P}, \mathcal{P}'}(\mathcal{S}) &= \sum_{P' \in \mathcal{P}'} |P'| \text{peak}\left(\frac{|U_{\mathcal{S}} \cap P'|}{|P'|}\right) \\
&= \sum_{P' \in \mathcal{P}'} |P'| \min\left\{\frac{|U_{\mathcal{S}} \cap P'|}{|P'|}, \frac{|U_{\mathcal{P} \setminus \mathcal{S}} \cap P'|}{|P'|}\right\} \\
&= \sum_{P' \in \mathcal{P}'} \min\{|U_{\mathcal{S}} \cap P'|, |U_{\mathcal{P} \setminus \mathcal{S}} \cap P'|\} \tag{4.4} \\
&\geq b(\mathcal{S}_s, \mathcal{S}_t). \tag{4.5}
\end{aligned}$$

□

We still have to make decisions on

1. the choice of the next part P from $\mathcal{P} \setminus (\mathcal{S}_s \dot{\cup} \mathcal{S}_t)$ that we use to extend either \mathcal{S}_s or \mathcal{S}_t and
2. whether we assign P to \mathcal{S}_s or \mathcal{S}_t .

Our strategy for item 2.) above is to assign P to \mathcal{S}_s or \mathcal{S}_t according to the (optimistic) prospect $b(\mathcal{S}_s, \mathcal{S}_t)$, i. e. P is assigned such that the new value $b(\mathcal{S}_s, \mathcal{S}_t)$ is minimal. We prefer minimizing $b(\mathcal{S}_s, \mathcal{S}_t)$ over minimizing $\phi_{\mathcal{P}, \mathcal{P}'}(\mathcal{S}_s)$ and/or $\phi_{\mathcal{P}, \mathcal{P}'}(\mathcal{S}_t)$ because the latter two numbers can both be high although the prospect for finding a good cut of \mathcal{P} is still good. Due to the symmetry of $\phi_{\mathcal{P}, \mathcal{P}'}(\cdot)$, however, the objectives of minimizing $b(\mathcal{S}_s, \mathcal{S}_t)$, $\phi_{\mathcal{P}, \mathcal{P}'}(\mathcal{S}_s)$ and $\phi_{\mathcal{P}, \mathcal{P}'}(\mathcal{S}_t)$ will have converged by the time when \mathcal{S}_s and \mathcal{S}_t are fully grown, i. e. $\mathcal{S}_s \dot{\cup} \mathcal{S}_t = \mathcal{P}$.

Our strategy for item 1.), i. e. the choice of P , aims at shifting the backtrack- ing phases of our branch-and-bound algorithm to scenarios in which \mathcal{S}_s and \mathcal{S}_t are already large and where chances are that we are close to a new minimum of $\phi_{\mathcal{P}, \mathcal{P}'}(\cdot)$. To this end, we pick P from $\mathcal{P} \setminus (\mathcal{S}_s \dot{\cup} \mathcal{S}_t)$ such that the alternative between putting P into \mathcal{S}_s or into \mathcal{S}_t matters the most in terms of $b(\cdot, \cdot)$. Formally,

$$P = \operatorname{argmax}_{P \in \mathcal{P} \setminus (\mathcal{S}_s \dot{\cup} \mathcal{S}_t)} |b(\mathcal{S}_s \dot{\cup} \{P\}, \mathcal{S}_t) - b(\mathcal{S}_s, \mathcal{S}_t \dot{\cup} \{P\})| \tag{4.6}$$

Algorithm 1 summarizes the way in which we extend a pair $(\mathcal{S}_s, \mathcal{S}_t)$.

Algorithm `greedy`($\mathcal{S}_s, \mathcal{S}_t, \text{bestSoFar}$) terminates prematurely, i. e. with $\mathcal{S}_s \dot{\cup} \mathcal{S}_t \neq \mathcal{P}$, if there is no chance to find \mathcal{S} with $\phi_{\mathcal{P}, \mathcal{P}'}(\mathcal{S}) < \text{bestSoFar}$. In the first call of `greedy`($\mathcal{S}_s, \mathcal{S}_t, \text{bestSoFar}$) we have $\mathcal{S}_s = \{P_s\}$, $\mathcal{S}_t = \{P_t\}$ and $\text{bestSoFar} = \infty$. In particular, `greedy`($\{P_s\}, \{P_t\}, \infty$) does *not* end prematurely, i. e. it delivers a P_s - P_t cut $(\mathcal{S}, \mathcal{P} \setminus \mathcal{S})$.

After initializing \mathcal{S}_s and \mathcal{S}_t , our branch-and-bound algorithm, see Algorithm 2, calls `greedy`($\mathcal{S}_s, \mathcal{S}_t, \infty$). In later calls of `greedy`($\mathcal{S}_s, \mathcal{S}_t, \text{bestSoFar}$) we always have $(\mathcal{S}_s \supsetneq \{P_s\} \vee \mathcal{S}_t \supsetneq \{P_t\}) \wedge \mathcal{S}_s \cap \mathcal{S}_t = \emptyset$, and bestSoFar amounts to the minimum weight ($\phi_{\mathcal{P}, \mathcal{P}'}$ value) of the P_s - P_t cuts found so far (see lines 5-10 of Algorithm 2). Two crucial questions *after* any call of `greedy` are

Algorithm 1 Algorithm $\text{greedy}(\mathcal{S}_s, \mathcal{S}_t, \text{bestSoFar})$ for extending a pair $(\mathcal{S}_s, \mathcal{S}_t)$ with $\mathcal{S}_s \cap \mathcal{S}_t = \emptyset$ towards a pair $(\mathcal{S}, \mathcal{P} \setminus \mathcal{S})$ with $\mathcal{S} \supseteq \mathcal{S}_s$ and $\mathcal{S} \cap \mathcal{S}_t = \emptyset$ as long as there is a chance that $\phi_{\mathcal{P}, \mathcal{P}'}(\mathcal{S}) < \text{bestSoFar}$.

```

1: while  $(\mathcal{S}_s \dot{\cup} \mathcal{S}_t \neq \mathcal{P}) \wedge b(\mathcal{S}_s, \mathcal{S}_t) < \text{bestSoFar}$  do
2:   Find  $P \in \mathcal{P} \setminus (\mathcal{S}_s \dot{\cup} \mathcal{S}_t)$  that fulfills Equation (4.6).
3:   if  $b(\mathcal{S}_s \dot{\cup} \{P\}, \mathcal{S}_t) < b(\mathcal{S}_s, \mathcal{S}_t \dot{\cup} \{P\})$  then
4:      $\mathcal{S}_s \leftarrow \mathcal{S}_s \dot{\cup} \{P\}$ 
5:   else
6:      $\mathcal{S}_t \leftarrow \mathcal{S}_t \dot{\cup} \{P\}$ 
7:   end if
8: end while

```

Algorithm 2 Branch-and-bound algorithm for finding a minimal P_s - P_t cut.

```

1:  $\mathcal{S}_s \leftarrow \{P_s\}, \mathcal{S}_t \leftarrow \{P_t\}$ 
2:  $\text{bestSoFar} \leftarrow \infty$ 
3: do
4:    $\text{greedy}(\mathcal{S}_s, \mathcal{S}_t, \text{bestSoFar})$ 
5:   if  $\mathcal{S}_s \dot{\cup} \mathcal{S}_t = \mathcal{P}$  then ▷ i. e. we have found a  $P_s$ - $P_t$  cut
6:     if  $b(\mathcal{S}_s, \mathcal{S}_t) < \text{bestSoFar}$  then ▷  $b(\mathcal{S}_s, \mathcal{S}_t) = \phi_{\mathcal{P}, \mathcal{P}'}(\mathcal{S}_s) = \phi_{\mathcal{P}, \mathcal{P}'}(\mathcal{S}_t)$ 
7:        $\mathcal{S} \leftarrow \mathcal{S}_s$ 
8:        $\text{bestSoFar} \leftarrow b(\mathcal{S}_s, \mathcal{S}_t)$ 
9:     end if
10:  end if
11:   $i \leftarrow 0$  ▷ Beginning of undo
12:  do
13:    if  $\hat{P}_{|\mathcal{S}_s \dot{\cup} \mathcal{S}_t| - 2 - i} \in \mathcal{S}_s$  then
14:       $\mathcal{S}_s \leftarrow \mathcal{S}_s \setminus \hat{P}_{|\mathcal{S}_s \dot{\cup} \mathcal{S}_t| - 2 - i}$ 
15:    else ▷ i. e.  $\hat{P}_{|\mathcal{S}_s \dot{\cup} \mathcal{S}_t| - 2 - i} \in \mathcal{S}_t$ 
16:       $\mathcal{S}_t \leftarrow \mathcal{S}_t \setminus \hat{P}_{|\mathcal{S}_s \dot{\cup} \mathcal{S}_t| - 2 - i}$ 
17:    end if
18:     $i \leftarrow i + 1$ 
19:  while  $(\mathcal{S}_s, \mathcal{S}_t) \neq (\{P_s\}, \{P_t\}) \wedge \text{dejaVu}(A(\mathcal{S}_s, \mathcal{S}_t))$  ▷ End of undo
20:  if  $(\mathcal{S}_s, \mathcal{S}_t) \neq (\{P_s\}, \{P_t\})$  then
21:     $(\mathcal{S}_s, \mathcal{S}_t) \leftarrow A(\mathcal{S}_s, \mathcal{S}_t)$ 
22:  end if
23: while  $(\mathcal{S}_s, \mathcal{S}_t) \neq (\{P_s\}, \{P_t\})$ 
24: return  $(\mathcal{S}, \mathcal{P} \setminus \mathcal{S})$ 

```

1) whether $\text{greedy}(\cdot, \cdot, \cdot)$ needs to be invoked again and, if so,

- 2a) which of the most recent assignments of parts (to \mathcal{S}_s or \mathcal{S}_t) should be undone in a backtracking step and
- 2b) which alternative line for searching a minimal P_s - P_t cut is taken after backtracking, i. e. what is the input of $\text{greedy}(\cdot, \cdot, \cdot)$ when it is called the next time.

The answer to 1) is “as long as $\mathcal{S}_s \supsetneq \{P_s\}$ or $\mathcal{S}_t \supsetneq \{P_t\}$ ”. In other words, we stop when $(\mathcal{S}_s, \mathcal{S}_t)$ has shrunk to its initialization $(\{P_s\}, \{P_t\})$ (see lines 3 and 23 of Algorithm 2). The following notation and definition will make it easier to answer the remaining questions.

Notation 4.1 *W.l.o.g. the parts in $\mathcal{S}_s \dot{\cup} \mathcal{S}_t \setminus \{P_s, P_t\}$ are denoted by $\hat{P}_1, \dots, \hat{P}_{|\mathcal{S}_s \dot{\cup} \mathcal{S}_t| - 2}$, and the indices reflect the order in which the parts were added to $\mathcal{S}_s \setminus \{P_s\}$ or $\mathcal{S}_t \setminus \{P_t\}$ (the larger an index, the later the part was added).*

Definition 7 (Alternative $A(\mathcal{S}_s, \mathcal{S}_t)$ to $(\mathcal{S}_s, \mathcal{S}_t)$).

If $\hat{P}_{|\mathcal{S}_s \dot{\cup} \mathcal{S}_t| - 2}$ is contained in \mathcal{S}_s , the alternative to $(\mathcal{S}_s, \mathcal{S}_t)$ is

$$(\mathcal{S}_s \setminus \{\hat{P}_{|\mathcal{S}_s \dot{\cup} \mathcal{S}_t| - 2}\}, \mathcal{S}_t \dot{\cup} \{\hat{P}_{|\mathcal{S}_s \dot{\cup} \mathcal{S}_t| - 2}\}).$$

If $\hat{P}_{|\mathcal{S}_s \dot{\cup} \mathcal{S}_t| - 2}$ is contained in \mathcal{S}_t , the alternative to $(\mathcal{S}_s, \mathcal{S}_t)$ is

$$(\mathcal{S}_s \dot{\cup} \{\hat{P}_{|\mathcal{S}_s \dot{\cup} \mathcal{S}_t| - 2}\}, \mathcal{S}_t \setminus \{\hat{P}_{|\mathcal{S}_s \dot{\cup} \mathcal{S}_t| - 2}\}).$$

The answer to 2a) now is “Undo the assignment of $\hat{P}_{|\mathcal{S}_s \dot{\cup} \mathcal{S}_t| - 2}$. Keep undoing the latest assignments until some $\hat{P}_{|\mathcal{S}_s \dot{\cup} \mathcal{S}_t| - 2 - i}$, $i \geq 1$, is reached such that $\text{greedy}(\cdot, \cdot, \cdot)$ has *not* yet been called with the first two arguments given by $A(\mathcal{S}_s, \mathcal{S}_t)$.” In the pseudocode of Algorithm 2, a boolean function called $\text{dejaVu}(\cdot, \cdot)$ is used to express whether $A(\mathcal{S}_s, \mathcal{S}_t)$ has entered the call of $\text{greedy}(\cdot, \cdot, \cdot)$ before, see line 19 of Algorithm 2. This line guarantees termination of our branch-and-bound algorithm. The answer to 2b) then is “call $\text{greedy}(\cdot, \cdot, \cdot)$ with $A(\mathcal{S}_s, \mathcal{S}_t)$ and the current value of bestSoFar ” (see lines 21 and 4 of Algorithm 2).

We conclude this section with two implementation details.

- We implement the boolean function $\text{dejaVu}(\cdot, \cdot)$ as a boolean vector called $\text{doneWith}[\cdot]$. The two arguments $\mathcal{S}_s, \mathcal{S}_t$ of $\text{dejaVu}(\cdot, \cdot)$ correspond to a single index i of doneWith . Specifically, i is the cardinality of $\mathcal{S}_s \dot{\cup} \mathcal{S}_t \setminus \{P_s, P_t\}$ or, equivalently, the highest index, $|\mathcal{S}_s \dot{\cup} \mathcal{S}_t| - 2$, in Notation 4.1. The vector doneWith has length $|\mathcal{P}| - 2$, and its entries are initialized to false. Anytime $\text{greedy}(\cdot, \cdot, \cdot)$ is called with $(\mathcal{S}_s, \mathcal{S}_t) = A(\hat{\mathcal{S}}_s, \hat{\mathcal{S}}_t)$ for some $\hat{\mathcal{S}}_s, \hat{\mathcal{S}}_t \subset \mathcal{P}$, $\text{doneWith}[i]$ is set to true. Furthermore, if backtracking goes back behind an index j , $\text{doneWith}[j]$ is set to false.
- For efficient updates of $b(\mathcal{S}_s, \mathcal{S}_t)$ in Algorithm 2, i.e. when \mathcal{S}_s or \mathcal{S}_t grows or shrinks by one part, we use distributions as in Section 3.3 — this time the *rows* of the contingency table of \mathcal{P} and \mathcal{P}' . Specifically, let $P \in \mathcal{P}$. Then $d_P[\cdot]$ is a vector of length $|\mathcal{P}'|$ defined by $d_P[j] := |P \cap P'_j|$ for $1 \leq j \leq |\mathcal{P}'|$. We define $d_{\mathcal{S}_s} := \sum_{P \in \mathcal{S}_s} d_P$. Now, adding or removing a part P from \mathcal{S}_s corresponds to the command $d_{\mathcal{S}_s} \leftarrow d_{\mathcal{S}_s} + d_P$ and $d_{\mathcal{S}_s} \leftarrow d_{\mathcal{S}_s} - d_P$, respectively. By carrying along $d_{\mathcal{S}_s}$ and $d_{\mathcal{S}_t}$, we can compute the bounds $b(\cdot, \cdot)$ as follows.

$$b(\mathcal{S}_s, \mathcal{S}_t) = \sum_{j=1}^{|\mathcal{P}'|} \min\{d_{\mathcal{S}_s}[j], d_{\mathcal{S}_t}[j]\}. \quad (4.7)$$

Such an implementation is also advantageous for parallel processing.

5 Experiments

The purpose of the experiments of Section 5.2 is to test the limits of the branch-and-bound algorithm described in Section 4.2. In Section 5.3 we use correspondences to gain insight into the behavior of a community detection algorithm and an extension thereof.

5.1 Settings and implementation

All computations were done on a workstation with two 8-core Intel(R) Xeon(R) E5-2680 processors at 2.7 GHz. Our C++ code was compiled with GCC 4.7.1, and we use Python 3.4.1.

We integrated our code into NETWORKIT, an open-source network analysis tool suite [23], because NETWORKIT provides (i) a C++ class for partitions, (ii) implementations of the community detection algorithm which we want to study and (iii) a Python frontend which allows us to conveniently include runs of community detection algorithms into our tests. The C++ sources will be made publicly available after paper acceptance.

5.2 Running times of branch-and-bound algorithm

The idea behind the experiments of this section is to see how well our branch-and-bound algorithm can cope with situations in which large subsets \mathcal{S} and \mathcal{S}' have to be formed to minimize $\phi_{\mathcal{P}, \mathcal{P}'}(\cdot, \cdot)$.

To this end we first generate a partition $\mathcal{P} = \{P_1, P_2, \dots, P_{2k}\}$ of a set V where all parts have the same size. Second, we randomly select $\mathcal{S} \subset \mathcal{P}$ with $|\mathcal{S}| = k$. Third, we generate $\mathcal{P}' = \{P'_1, P'_2, \dots, P'_{2k}\}$ with empty parts and randomly select $\mathcal{S}' \subset \mathcal{P}'$ with $|\mathcal{S}'| = k$. Finally, we fill the parts of \mathcal{P}' as follows. The elements in \mathcal{S} [$\mathcal{P} \setminus \mathcal{S}$] are distributed evenly but otherwise randomly over the parts in \mathcal{S}' [$\mathcal{P}' \setminus \mathcal{S}'$]. Thus, $(\mathcal{S}, \mathcal{S}')$ is a perfect but well-hidden correspondence with large $\mathcal{S}, \mathcal{S}'$, and the distribution of V over \mathcal{P}' is random otherwise. For a small example see Figure 5.

In our experiments we choose $2k \in \{10, 20, 50, 100, 200, 400, 800, 1000\}$ and $|V| = 10^7$. We then calculate the best $2k - 1$ correspondences via a Gomory-Hu tree of minimal P_s - P_t cuts of \mathcal{P} w.r.t. \mathcal{P}' . We use Gusfield's algorithm for the computation of the Gomory-Hu tree [11, 9]. The running times for calculating the best $2k - 1$ correspondences are given in Figure 6. A moderate increase in running time to about $2k = 100$ is followed by a steep increase, which is expected for a branch-and-bound algorithm. In the experiments described in the next section, we observe very similar dependencies of the running time on the number of parts, not only qualitatively but also in terms of absolute running times.

The size of V is not a problem since it enters the running time only as a linear add-on (when the distributions are computed). Generally, our branch-and-bound

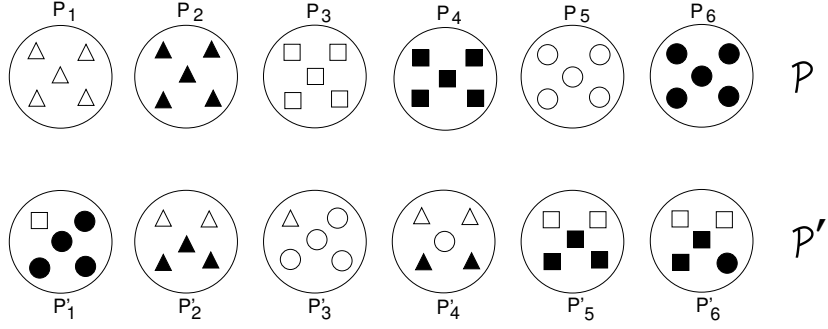


Fig. 5. Partitions \mathcal{P} and \mathcal{P}' are such that $\mathcal{S} = (\{P_1, P_2, P_5\}, \{P'_2, P'_3, P'_4\})$ is a perfect correspondence between \mathcal{P} and \mathcal{P}' .

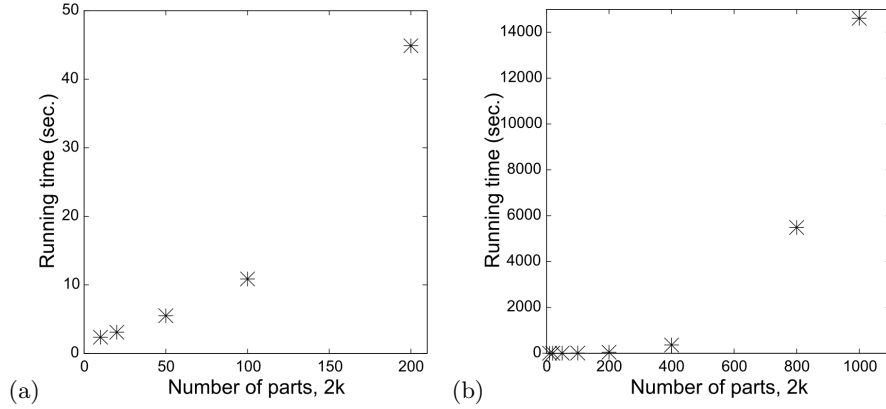


Fig. 6. Running times for calculating the $2k - 1$ best correspondences. (a) $2k = 10$ to $2k = 200$. (b) $2k = 10$ to $2k = 1000$.

algorithm can deal with partitions \mathcal{P} and \mathcal{P}' of very large data sets as long as the number of parts in \mathcal{P} and \mathcal{P}' is bounded by about 1000. As we have observed in additional experiments, the influence of $|\mathcal{P}'|$ on the running time is only linear. This makes sense since $|\mathcal{P}'|$ equals the length of the distributions, and \mathcal{P}' affects our branch-and-bound algorithm only via the distributions.

Finding correspondences may become easier in applications where domain knowledge can be used to steer the branch-and-bound algorithm. In image segmentation, for example, one can use color, shape, and connectivity to improve the choice of the next region during the extension of \mathcal{S}_s and \mathcal{S}_t in line 2 of Algorithm 1.

Table 5.1. Complex networks used for comparing partitions.

Graph ID	Name	#vertices	#edges	Network Type
1	P2P-GNUTELLA	6 405	29 215	filesharing network
2	PGPGIANTCOMPO	10 680	24 316	network of PGP users
3	EMAIL-EUALL	16 805	60 260	network of connections via email
4	AS-22JULY06	22 963	48 436	autonomous systems in the internet
5	SOC-SLASHDOT0902	28 550	379 445	news network
6	LOC-BRIGHTKITE_EDGES	56 739	212 945	location-based friendship network
7	LOC-GOWALLA_EDGES	196 591	950 327	location-based friendship network
8	COAUTHORS_CITeseer	227 320	814 134	citation network
9	WIKI-TALK	232 314	1 458 806	user interactions through edits
10	CITATION_CITeseer	268 495	1 156 647	citation network
11	COAUTHORSDBLP	299 067	977 676	citation network
12	WEB-GOOGLE	356 648	2 093 324	hyperlink network of web pages
13	CO PAPERS_CITeseer	434 102	16 036 720	citation network
14	CO PAPERSDBLP	540 486	15 245 729	citation network
15	AS-SKITTER	554 930	5 797 663	network of internet service providers

5.3 Behavior of a parallel implementation of the Louvain method for community detection in networks: insight through correspondences

In the field of *community detection in networks* one thinks of communities as densely connected sets of nodes with sparse connections to the rest of the network. Their detection has received significant attention in basic research and applications. For surveys see [19,5]. Here, we consider only non-overlapping communities that form partitions. One of the most popular methods in community detection is the method that we describe next.

The Louvain method (LM). LM is a locally greedy, bottom-up multilevel algorithm [3] that forms communities of network nodes. On each level, it assigns (i. e. moves) nodes to communities (parts) in an iterative way that maximizes the objective function modularity [7] (not to be confused with Definition 2). The communities found on a level are contracted into single nodes, giving rise to the graph on the next level. The communities of the coarsest graph are then successively expanded to the next finer level until, on the finest level, we have the final result.

In [22] a shared-memory parallelization of LM, called PLM, is provided: among other things, node moves are evaluated and performed in parallel instead of sequentially. We denote the single-thread (sequential) version of PLM by SLM. Moreover, PLM and thus SLM have been extended by an optional *refinement* phase: after each expansion nodes are again moved for modularity gain. SLM with refinement is denoted by SLMR.

Questions on behavior of PLM, SLM and SLMR. PLM is not deterministic due to race conditions during the parallel execution of the method. Let \mathcal{P} be the

partition returned by one run of PLM, and let \mathcal{P}' be the partition returned by another run of PLM. We want to know whether the transition from \mathcal{P} to \mathcal{P}' is best described as (a) communities merely exchanging elements with each other, but otherwise remaining as they are or (b) involving unions and break-ups of communities. An analogous question arises when \mathcal{P} and \mathcal{P}' are from SLM and SLMR, respectively.

Approach. As input for PLM, SLM, and SLMR we choose a collection of 15 diverse and widely used complex networks from two popular archives [1,14]. These networks are listed and described in Table 5.1. For each network and each comparison, i.e. one run of PLM vs. another run of PLM or SLM vs. SLMR, we get a pair of partitions \mathcal{P} and \mathcal{P}' .

We first want to get an idea of the overall dissimilarity of the correspondences between \mathcal{P} and \mathcal{P}' . To this end we add up the $\phi_{\mathcal{P},\mathcal{P}'}(\cdot,\cdot)$ values of the $|\mathcal{P}|-1$ best correspondences (those that are given by the fundamental cuts of the Gomory-Hu tree) and divide this sum by the total number of vertices in the graph.

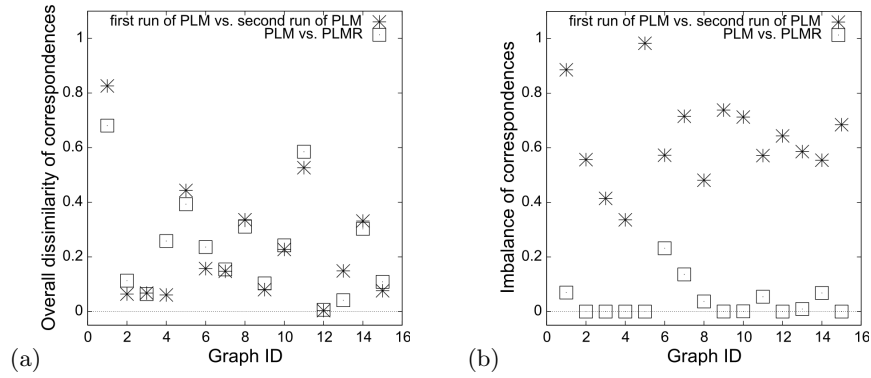


Fig. 7. (a) Overall dissimilarities and (b) imbalances of correspondences for each graph in Table 5.1.

We then tackle the question about the occurrence of unions and break-ups of communities through computing pairs $(|\mathcal{S}|, |\mathcal{S}'|)$: one pair for each of the $|\mathcal{P}|-1$ best correspondences $(\mathcal{S}, \mathcal{S}')$, see Figure 8. We distinguish between pairs in $M_{eq} := \{(n, n') : n = n' \in \mathbb{N}\}$ and the remaining pairs in $M_{neq} := \mathbb{N} \times \mathbb{N} \setminus M_{eq}$. Let ϕ_{eq} [ϕ_{neq}] denote the sum over all $\phi_{\mathcal{P},\mathcal{P}'}(\cdot,\cdot)$ values (overall dissimilarity) of correspondences with pairs in M_{eq} [M_{neq}]. We define the *imbalance* of the correspondences between \mathcal{P} and \mathcal{P}' as $\frac{\phi_{neq}}{\phi_{eq} + \phi_{neq}}$. Thus, a low [high] imbalance indicates that the communities tend to remain intact [unions and break-ups of communities are frequent].

Results. Figure 7a shows that the overall dissimilarity of the correspondences between different runs of PLM, i.e. the median of overall dissimilarity from 10

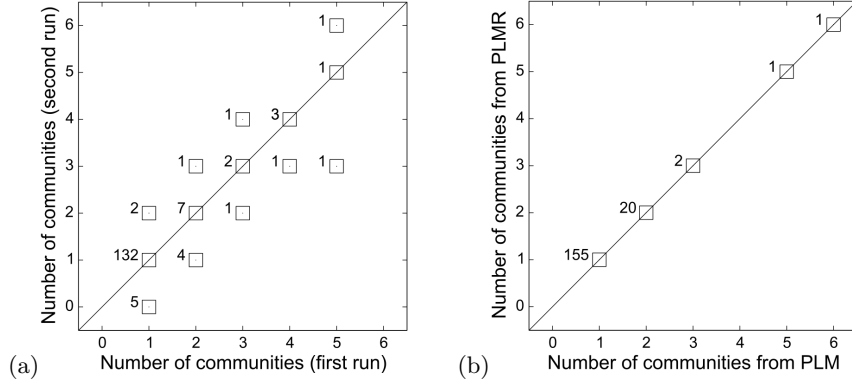


Fig. 8. A data point (n, n') indicates that there are correspondences between partitions \mathcal{P} and \mathcal{P}' , where n communities of \mathcal{P} correspond to n' communities of \mathcal{P}' . The number at a data point (n, n') indicates the number of correspondences $(\mathcal{S}, \mathcal{S}')$ with $(|\mathcal{S}|, |\mathcal{S}'|) = (n, n')$. PLM, SLM and SLMR are applied to the graph WEB-GOOGLE, see Table 5.1. (a) Pairs from different runs of PLM. The pairs not shown are $(37, 36)$ and $(94, 94)$ with one occurrence each. (b) Pairs from SLM vs. SLMR. The pairs not shown are $(61, 61)$, $(66, 66)$, $(69, 69)$ and $(77, 77)$ with one occurrence each.

experiments with two runs each, is on the same order as the overall dissimilarity of the correspondences between SLM and SLMR.

Figure 7b shows the imbalances. For the comparison PLM vs. PLM, the imbalances are again medians over 10 experiments with two runs each. We see that, despite comparable overall dissimilarity of the correspondences, the imbalances from different runs of PLM are much higher than those from SLM vs. SLMR. In the latter comparison, correspondences from graphs with IDs 2, 4, 5, 12 and 15 have imbalance exactly zero, i. e. *all* correspondences are between the same number of communities.

The comparison PLM vs. PLM is characterized by high imbalances *with no exception*. Even for the graph WEB-GOOGLE (with ID 12), where the overall dissimilarity of the correspondences is very small in both comparisons, i. e. 0.0033 in PLM vs. PLM and 0.0065 in SLM vs. SLMR, the latter dissimilarity being about twice the former, the correspondences in the comparison SLM vs. SLMR are definitely much more balanced. See Figures 7b and 8.

To summarize, the imbalances we gathered indicate that the race conditions during PLM disrupt the communities in a more fundamental way (frequent unions or break-ups of communities) than the refinement phase. The latter deserves its name in that it leaves most communities intact. Yet, the impact of the race conditions is comparable to the impact of the refinement if impact is measured by $\phi_{\mathcal{P}, \mathcal{P}'}(\cdot, \cdot)$: communities do unite and break up due to race conditions, but primarily those communities that are not distinct in the first place.

6 Related work

Similarity measures for partitions. Wagner and Wagner [27] provide a comprehensive collection of similarity measures for partitions $\mathcal{P} = \{P_1, \dots, P_{|\mathcal{P}|}\}$ and $\mathcal{P}' = \{P'_1, \dots, P'_{|\mathcal{P}'|}\}$ of the same set V . They can all be derived from the contingency table of \mathcal{P} and \mathcal{P}' . The correspondences introduced in this paper can also be derived from the contingency table. Wagner and Wagner group the similarity measures into three groups:

1. Measures based on considering all unordered pairs (two-element subsets) $\{v, w\}$ of V and counting the 4 cases arising from the distinction as to whether v and w belong to same part or to different parts of \mathcal{P} and the analogous distinction with \mathcal{P}' instead of \mathcal{P} . Examples of such measures are the Rand index [18] and the adjusted Rand index [12].
2. Measures that involve a sum over maximum P_i, P'_j overlaps, where the sum is over the P_i , the maximum is over the P'_j , and the overlaps are defined in various ways. One example is the \mathcal{F} -measure [13,6]. Typically, these measures yield different results if the roles of \mathcal{P} and \mathcal{P}' are exchanged. Fragmentation defined in this paper, see Equations (1.3) and (1.4), has similar properties in that it (i) involves a sum over the P'_j , (ii) aggregates P_i, P'_j overlaps over certain P_i in a nonlinear way, and (iii) may vary if the roles of \mathcal{P} and \mathcal{P}' are exchanged.
3. Measures that involve mutual information, e. g. Normalized Mutual Information [25]. Here, the common ground with our approach to defining fragmentation and thus correspondences is that we can replace the function $\text{peak}(\cdot)$, see Equations (1.3) and (1.4), by the binary entropy function without altering the nature of our optimization problem (see also the next paragraph and Section 3.1).

Impurity measures. The fragmentation of $\mathcal{S} \subseteq \mathcal{P}$ w. r. t. \mathcal{P}' , i. e. $\phi_{\mathcal{P}, \mathcal{P}'}(\mathcal{S})$ in the form of Equations (1.3) and (1.4), indicates how well the parts of \mathcal{P}' fit into $U_{\mathcal{S}}$ or $V \setminus U_{\mathcal{S}}$. Impurity measures, as defined in [26,21], seem to be based on a similar idea. Using our setting and notation, Simovici *et al.* [21] define the impurity of a subset L of the ground set V relative to \mathcal{P} and generated by $\text{peak}(\cdot)$ as

$$\text{IMP}_{\mathcal{P}'}^{\text{peak}}(L) = |L| \sum_{P' \in \mathcal{P}'} \text{peak}\left(\frac{|L \cap P'|}{|L|}\right). \quad (6.1)$$

We can turn $\text{IMP}_{\mathcal{P}'}^{\text{peak}}(U_{\mathcal{S}})$ into $\phi_{\mathcal{P}, \mathcal{P}'}(\mathcal{S})$ by (i) pulling $U_{\mathcal{S}}$ under the sum (mathematically correct) and (ii) exchanging the roles of $U_{\mathcal{S}}$ and P' under the sum (mathematically incorrect). For us it is important to have the roles as they are in $\phi_{\mathcal{P}, \mathcal{P}'}(\mathcal{S})$ because this is what makes $\phi_{\mathcal{P}, \mathcal{P}'}(\cdot)$ a submodular and symmetric function. These properties, in turn, make it possible to find the best nontrivial \mathcal{S} in polynomial time.

Despite this mismatch between $\text{IMP}_{\mathcal{P}'}^{\text{peak}}(\cdot)$ and $\phi_{\mathcal{P}, \mathcal{P}'}(\cdot)$, studying $\text{IMP}_{\mathcal{P}'}^{\text{peak}}(\cdot)$ helped to develop the intuition behind our approach.

The main properties of $\text{IMP}_{\mathcal{P}'}^{\text{peak}}(\cdot)$ and related measures, as formulated and proven in [21], are preserved if $\text{peak}(\cdot)$ is replaced by another *generator*, as defined in [21], i.e. another concave and additive function $f : [0, 1] \mapsto \mathbb{R}$ with $f(0) = f(1) = 0$. Likewise, if we replace $\text{peak}(\cdot)$ in Equation (1.3) by another generator, we will arrive at similar definitions of fragmentation and correspondences. This also does not change the kind and asymptotic complexity of the optimization problems posed by our approach. Examples of generators are the binary entropy function and the Gini impurity measure [26, 21].

7 Conclusions and outlook

Correspondences and associated dissimilarities provide insight into multiple ways in which two partitions can be similar and are an instrument to detect, represent and quantify consensus between partitions. Very often, it is not clear in a certain application whether two or more parts “belong together” or should be single parts. Then, correspondences allow us to compare two partitions modulo such ambiguities.

Since “good” correspondences between two partitions \mathcal{P} and \mathcal{P}' are basically small cuts of \mathcal{P} w.r.t. \mathcal{P}' , a hierarchy of the $|\mathcal{P}| - 1$ best correspondences arises from a minimal basis of non-crossing P_s - P_t cuts of \mathcal{P} w.r.t. \mathcal{P}' . In particular, a hierarchy of the $|\mathcal{P}| - 1$ best correspondences can readily be derived from a Gomory-Hu tree, the computation of which requires the determination of only $|\mathcal{P}| - 1$ minimal P_s - P_t cuts. Our branch-and-bound algorithm makes it feasible to compute these cuts even for $|\mathcal{P}|$ and $|\mathcal{P}'|$ around 1000. The number of elements in V can be neglected. Within the branch-and-bound algorithm, the most crucial feature is the choice of the next candidates for extension of either \mathcal{S}_s or \mathcal{S}_t , as specified in Equation (4.6). The choice of the next candidates may also involve application-specific criteria such as color or shape in image segmentation. Such additional information may help our branch-and-bound algorithm to stay in the lane, and is expected to accelerate it.

The $|\mathcal{P}| - 1$ best correspondences between \mathcal{P} and \mathcal{P}' give rise to consensus partitions (clusterings) of \mathcal{P} and \mathcal{P}' (consensus modulo unions of parts). Indeed, any correspondence $(\mathcal{S}, \mathcal{S}')$ with $\phi_{\mathcal{P}, \mathcal{P}'}(\mathcal{S}, \mathcal{S}') = 0$ is such that each $P' \in \mathcal{P}'$ is fully contained in \mathcal{S} or in $\mathcal{P} \setminus \mathcal{S}$. Thus, removing the edges of the Gomory-Hu tree with weight zero yields the finest partition that is a coarsening of \mathcal{P} and \mathcal{P}' . In case of PLM vs. PLM [SLM vs. SLMR] on the network WEB-GOOGLE, the number of clusters in this “largest common denominator of \mathcal{P} and \mathcal{P}' ” is 54% [32%] of the number of clusters from PLM [SLM]. If we consider all correspondences $(\mathcal{S}, \mathcal{S}')$ whose quality $\frac{|U_{\mathcal{S}} \cap U_{\mathcal{S}'}|}{|U_{\mathcal{S}} \cup U_{\mathcal{S}'}|}$ exceeds 0.99, the percentage rises to 96% [68%]. Thus, the exact consensus already contains a considerable amount of clusters, and a very good consensus contains the majority of clusters. More generally, the $|\mathcal{P}| - 1$ best correspondences give rise to a hierarchy of consensus partitions parameterized by quality. Consensus partitions with more parts can also be obtained by generating finer partitions in the first place.

Despite the symmetry of $\phi_{\mathcal{P}, \mathcal{P}'}(\mathcal{S}, \mathcal{S}') = |U_{\mathcal{S}} \Delta U_{\mathcal{S}'}|$, finding the $|\mathcal{P}| - 1$ best correspondences between \mathcal{P} and \mathcal{P}' is not the same as finding the $|\mathcal{P}'| - 1$ best correspondences between \mathcal{P}' and \mathcal{P} , even if $|\mathcal{P}| = |\mathcal{P}'|$. Interestingly, the time complexity for finding the $|\mathcal{P}| - 1$ best correspondences between \mathcal{P} and \mathcal{P}' , i.e. $\mathcal{O}(|V| + |\mathcal{P}|^5 |\mathcal{P}'|)$, depends only linearly on $|\mathcal{P}'|$. This linear dependence goes back to using distributions, where each part P of \mathcal{P} gives rise to a distribution of P w.r.t. \mathcal{P}' , analogous to Definition 3. In the future we would like to find out whether we can use distributions w.r.t. \mathcal{P}' and \mathcal{P} to speed up the computation of correspondences.

References

- [1] D. A. Bader, H. Meyerhenke, P. Sanders, C. Schulz, A. Kappes, and D. Wagner. Benchmarking for graph clustering and partitioning. In *Encyclopedia of Social Network Analysis and Mining*, pages 73–82. 2014.
- [2] J. Bilmes. Submodularity Functions, Optimization, and Application to Machine Learning. Lecture at University of Washington, Seattle, 2012.
- [3] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.
- [4] Brian Everitt, editor. *Cluster analysis*. Wiley series in probability and statistics. Wiley, Chichester, 5th ed. edition, 2011.
- [5] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5):75 – 174, 2010.
- [6] B. C. M. Fung, K. Wang, and M. Ester. Hierarchical document clustering using frequent items. In *Proceedings of the SIAM International Conference on Data Mining*, 2003.
- [7] M. Girvan and M.E.J. Newman. Community structure in social and biological networks. *Proc. of the National Academy of Sciences*, 99(12):7821, 2002.
- [8] M. X. Goemans and V. S. Ramakrishnan. Minimizing submodular functions over families of sets. *COMBINATORICA*, 15(4):499–513, 1995.
- [9] A. V. Goldberg and K. Tsoutsoulis. Cut tree algorithms: An experimental study. *Journal of Algorithms*, 38(1):51–83, 2001.
- [10] R. E. Gomory and T. C. Hu. Multi-terminal network flows. *Journal of the Society for Industrial and Applied Mathematics*, 9(4):551–570, 1961.
- [11] D. Gusfield. Very simple methods for all pairs network flow analysis. *SIAM J. Comput.*, 19(1):143–155, 1990.
- [12] L. Hubert and P. Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.
- [13] B. Larsen and C. Aone. Fast and effective text mining using linear-time document clustering. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '99, pages 16–22, New York, NY, USA, 1999. ACM.

- [14] J. Leskovec. Stanford large network dataset collection. At <http://snap.stanford.edu/data/>, October 2011.
- [15] D. J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.
- [16] M. Meilă. Comparing clusterings. Technical Report 418, University of Washington, COLT, 2003.
- [17] M. Queyranne. Minimizing symmetric submodular functions. *Mathematical Programming*, 82(1-2):3–12, 1998.
- [18] W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.
- [19] Satu Elisa Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, 2007.
- [20] A. Schrijver. *Combinatorial Optimization - Polyhedra and Efficiency*. Springer, 2003.
- [21] D. A. Simovici, D. Cristofor, and L. Cristofor. Impurity measures in databases. *Acta Informatica*, 28:200–2, 2002.
- [22] C. Staudt and H. Meyerhenke. Engineering parallel algorithms for community detection in massive networks. *Parallel and Distributed Systems, IEEE Transactions on*, PP(99):1–1, 2015. Available online, to appear in printed form.
- [23] C. L. Staudt, Sazonovs A., and H. Meyerhenke. Networkkit: An interactive tool suite for high-performance network analysis. *CoRR*, abs/1403.3005, 2014.
- [24] P. Stobbe and A. Krause. Efficient minimization of decomposable submodular functions. In *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010, Vancouver, British Columbia, Canada.*, pages 2208–2216, 2010.
- [25] A. Strehl and J. Ghosh. Cluster ensembles — a knowledge reuse framework for combining multiple partitions. *J. Mach. Learn. Res.*, 3:583–617, 2003.
- [26] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.
- [27] S. Wagner and D. Wagner. Comparing Clusterings – An Overview. Technical Report 2006-04, Universität Karlsruhe (TH), 2007.